# 目　录

# 第 1 章　Java 语言基础

**本章导读**

Java 语言是一种功能强大的面向对象程序设计语言。多年来，Java 语言以其独有的特点及与网络的紧密结合，成为当今应用最广泛的计算机编程语言。本章主要介绍 Java 语言的产生和发展过程、Java 语言的特点和优越性、Java 的体系结构、Java 程序的开发环境及其基本结构和开发流程。

**知识要点**

- Java 的特点
- Java 的平台无关性
- JDK 的安装和环境变量的配置
- Java 程序的开发流程



## 1.1　软件开发原理及流程

学习程序设计语言之前，需要对软件运行原理有所理解，并对软件开发过程有所了解，这对于初学者尤为重要。

### 1.1.1　软件运行原理

计算机是人类 20 世纪最伟大、最重要的发明之一，其伟大之处就在于它能够以惊人的效率和前所未有的智能化来辅助人们更好地完成认识自然和改造自然的工作。它是有史以来第一种能够完成真正意义上的复杂的“学习”功能的机器，这就使得它具有了某种更接近人类的“思考”能力；与此同时，计算机所特有的超人的计算能力可以把人们的工作效率和生产效率提升成千上万倍，从而把人从最直接、最原始的生产第一线上解放出来，转而从事使用和操纵计算机的工作。

计算机是由不同部分组成的非常复杂的系统，如果把它比作是一项工作，硬件工程师将负责其身体各部分健康、完好；软件工程师将教会它如何学习和工作；计算机的操纵人员将向这个身体健康并学有所长的“工人”布置任务并监督其保质保量地完成。换句话说，计算机由硬件工程师赋予生命，由软件工程师注入灵魂，并最终在千千万万的操作人员手中发挥威力和作用。计算机由中央处理单元、算术逻辑单元、内存单元、输入单元、输出单元和外存单元组成。计算机的基本原理是存储程序和控制程序，开发人员预先把指挥计算机如何进行操作的指令序列(称为程序)和原始数据通过输入设备输送到计算机内存储器中。每一条指令中明确规定了计算机从哪个地址取数，进行什么操作，然后送到什么地址去等步骤。软件在运行之前将指令保存到内存中的过程称为内存加载或调入内存，这个内存加载的步骤是由 CPU 执行的。加载成功之后，CPU 将从内存中依次取出该软件程序的每一条指令并顺序执行。在执行过程中，CPU 可能需要内存中这个软件或其他软件的数据，可能需要调动输入、输出单元完成输入、输出操作，也可能要调度它的软件指令配合工作。这一切，都取决于开发人员事先编写好并已经加载到内存中的程序指令。计算机系统软件运行的基本原理如图 1-1 所示。

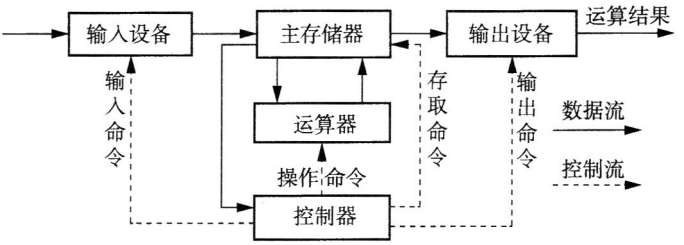



图 1-1 计算机工作原理图

程序设计语言是软件开发人员与计算机进行沟通和交流的工具，是计算机能够识别的语言。只有掌握了程序设计语言，软件开发人员才能指挥计算机按照自己的意志完成种种复杂的工作。

## 1.1.2 软件开发流程

由于软件系统被划分成操作系统软件、系统软件和应用软件，从事软件开发的人员也进行了相应的分工。操作系统软件是硬件裸机和其他软件或用户之间的必由接口，它的性能将决定整个计算机系统的性能，所以其开发要求很高，需要精深的专业知识与技能，正因如此，相应的从业人员也最少。系统软件是操作系统软件和应用软件之间的接口，从事系统软件开发一方面需要开发人员对操作系统有足够深入的了解，以便能充分利用操作系统提供的服务；另一方面，系统软件自身也需要为其上的应用软件提供方便、充分的服务，使应用软件可以不必了解操作系统的细节而直接使用系统软件的功能，从事系统软件开发的人员也较少。应用软件针对某个具体问题或实体，所以功能的专用性最强，软件间的差异性最大，开发的需求量最大，从业人员的人数也最多。开发操作系统软件或系统软件多注重于软件

的性能、效率，而开发应用软件则注重用户的需求，即充分研究应用软件的最终用户和操作者希望这个软件具有何种功能，能解决何种问题，并在明确需求的基础之上再去寻找一个能满足这个需求的解决方案：是直接将系统建筑在操作系统之上，还是寻找一个合适的系统软件作为基础，选择何种计算结构等。无论何种情况，开发人员都需要对即将研发的应用软件所立足的基础层次有足够的了解，并掌握这个层次的相应开发工具，为此，我们需要对软件开发流程有所了解。

软件开发流程(Software Development Process)即软件设计思路和方法的一般过程，包括设计软件的功能和实现的算法和方法、软件的总体结构设计和模块设计、编程和调试、程序联调和测试以及编写、提交程序。软件开发流程如图 1-2 所示。

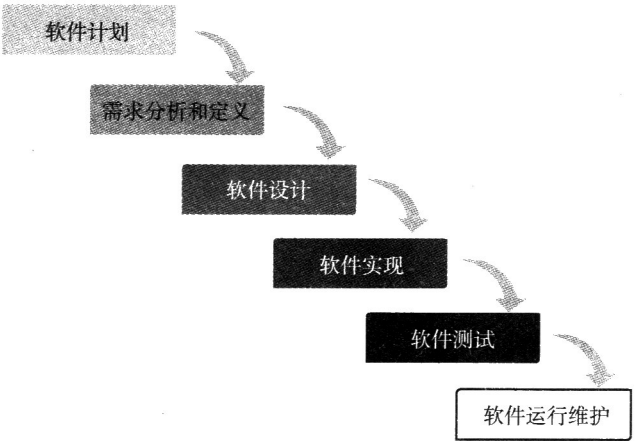



**图 1-2　软件开发流程**

1. 需求调研分析

相关系统分析员和用户初步了解需求，然后用 Word 列出要开发的系统的大功能模块，以及每个大功能模块有哪些小功能模块。对于有些需求比较明确的相关界面，在这一步里面可以初步定义好少量的界面。

系统分析员深入了解和分析需求，根据自己的经验和需求用 Word 或相关的工具再做出一份系统的功能需求文档。这次的文档会清楚地列出系统大致的大功能模块，大功能模块有哪些小功能模块，并且还列出相关的界面和界面功能。

系统分析员和用户再次确认需求。

2. 概要设计

首先，开发者需要对软件系统进行概要设计，即系统设计。概要设计需要对软件系统的设计进行考虑，包括系统的基本处理流程、系统的组织结构、模块划分、功能分配、接口设计、运行设计、数据结构设计和出错处理设计等，从而为软件的详细设计提供基础。

3. 详细设计

在概要设计的基础上，开发者需要进行软件系统的详细设计。详细设计需要描述实现具体模块所涉及的主要算法、数据结构、类的层次结构及调用关系，需要说明软件系统各个

层次中的每一个程序(每个模块或子程序)的设计考虑,以便进行编码和测试,还应当保证软件的需求完全分配给整个软件。详细设计应当足够详细,使开发者能够根据详细设计报告进行编码。

4. 编码

在软件编码阶段,开发者根据《软件系统详细设计报告》中对数据结构、算法分析和模块实现等方面的设计要求,开始具体的程序编写工作,分别实现各模块的功能,从而实现对目标系统的功能、性能、接口、界面等方面的要求。

5. 测试

测试编写好的系统。交付给用户使用,用户使用后一一地确认每个功能。

6. 软件交付准备

在软件测试证明软件达到要求后,软件开发者应向用户提交开发的目标安装程序、数据库的数据字典、《用户安装手册》、《用户使用指南》、需求报告、设计报告、测试报告等双方合同约定的内容。《用户安装手册》应详细介绍安装软件对运行环境的要求、安装软件的定义和内容、安装软件在客户端和服务器端及中间件的具体安装步骤、软件安装后的系统配置。《用户使用指南》应包括软件各项功能的使用流程、操作步骤、相应业务介绍、特殊提示和注意事项等方面的内容,在需要时还应举例说明。

7. 验收

用户验收。

### 1.1.3 程序设计语言

程序设计语言是能够被计算机和编程人员双方所理解和认可的交流工具。当软件开发人员希望计算机完成一件工作,或解决一个问题时,他(她)首先需要把这个问题的实质彻底研究清楚,确定解决问题的方法和步骤;然后再把这个方法和步骤用计算机能够理解和执行的程序设计语言表述出来,形成一组语句的集合,即程序。

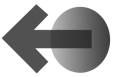
程序设计语言并不唯一,在计算机技术发展的 50 年中,先后形成了数百种不同的程序设计语言。按照其发展历史,程序设计语言按其级别可以划分为机器语言、汇编语言和高级语言、第四代语言四大类。

1. 第一代语言(机器语言)

机器语言就是计算机的指令系统,用机器语言编写的程序可以被计算机直接执行。由于不同类型计算机的指令系统(机器语言)不同,因而在一种类型计算机上编写的机器语言程序,在另一种类型的计算机上也可能无法运行。机器语言程序全部用二进制(八进制、十六进制)代码编制,人们不易记忆和理解,也难于修改和维护,所以现在已不用机器语言编制程序了。

2. 第二代语言(汇编语言)

汇编语言用助记符来代替机器指令的操作码和操作数,如用 ADD 表示加法、SUB 表示



减法、MOV 表示传送数据等。这样就能使指令使用符号表示而不再使用二进制表示。用汇编语言编写的程序与机器语言程序相比，虽然可以提高一点效率，但仍然不够直观简便。

3. 第三代语言(高级语言)

高级语言是面向用户的、基本上独立于计算机种类和结构的语言。其最大的优点是：形式上接近于算术语言和自然语言，概念上接近于人们通常使用的概念。高级语言的一个命令可以代替几条、几十条甚至几百条汇编语言的指令。因此，高级语言易学易用、通用性强、应用广泛。从描述客观系统来看，程序设计语言可以分为面向过程语言和面向对象语言。

(1)面向过程语言

以“数据结构＋算法”程序设计范式构成的程序设计语言，称为面向过程语言。前面介绍的程序设计语言大多为面向过程语言。

(2)面向对象语言

以“对象＋消息”程序设计范式构成的程序设计语言，称为面向对象语言。目前比较流行的面向对象语言有 Delphi、Java 和 C＋＋等。

4. 第四代语言(简称 4GL)

4GL 是非过程化语言，编码时只需说明“做什么”，不需描述算法细节。数据库查询是 4GL 的典型应用。用户可以用结构化查询语言(SQL)对数据库中的信息进行复杂的操作，如用户只需将要查找的内容在什么地方、根据什么条件进行查找等信息告诉 SQL，SQL 将自动完成查找过程。第四代程序设计语言是面向应用，为最终用户设计的一类程序设计语言。它具有缩短应用开发过程、降低维护代价、最大限度地减少调试过程中出现的问题以及对用户友好等优点。



## 1.2　Java 语言简介

Java 语言是一种简单的，面向对象的，分布式的，具备解释性、健壮性、安全与系统无关性、可移植性的，高性能、多线程的动态语言。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于 PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网。在全球云计算和移动互联网的产业环境下，Java 更具备了显著优势和广阔前景。

### 1.2.1　Java 语言的发展

Java 的前身是 Sun MicroSystems 公司开发的一种用于智能化家电的名为“橡树”(Oak)的语言。Oak 语言当时几近失败，直到 1993 年才随着 WWW(万维网)的迅速发展而重现生机。Sun 公司发现可以利用这种技术创造含有动态内容的 WWW 网页，便组织人力对其进行重新开发和改造。1995 年 5 月 23 日，Java 这种定位于网络应用的程序设计语言被正式推出，自那

以后,Java 逐步从一种单纯的高级编程语言发展成为一种重要的基于 Internet 的开发平台,并进而带动了 Java 产业的发展和壮大,成为当今计算机不可忽视的力量和重要的发展潮流。

### 1.2.2 Java 语言的组成

Java 语言由语法规则和类库两部分组成。语法规则确定了 Java 程序的书写规范;类库,或称为运行时库,则提供了 Java 程序与运行它的系统软件(Java 虚拟机)之间的接口。如果把用 Java 语言编写的程序看成是我们前面讨论过的应用软件,那么 Java 类库就是支持这种应用软件的系统软件的一部分。它实际上是一组由其他开发人员或软件供应商编写好的 Java 程序模块,每个模块通常对应一种特定的基本功能和任务,这样当我们自己编写的 Java 程序需要完成其中某一功能的时候,就可以直接利用这些现成的类库,而不需要从头编写。

学习 Java 语言程序设计,也相应地要把注意力集中在两个方面:一是其语法规则,这是编写 Java 程序的基本功;另一个是类库,这是提高编程效率和质量的必由之路,甚至从一定程度上来说,能否熟练自如地掌握尽可能多的 Java 类库决定了一个 Java 程序员编程能力的高低。同时,Java 作为一门网络应用语言,其程序的计算结构相对于以往其他单机上工作的程序也更为复杂。充分了解这些复杂环境和结构,也是对 Java 开发人员的基本要求。

### 1.2.3 Java 语言的版本

1999 年 6 月,Sun 公司发布. Java 的 3 个版本:

标准版(J2SE-Java2 Platform,Standard Edition):提供基础 Java 开发工具、执行环境与 API。

企业版(J2EE-Java2,Platform,Enterprise Edition):由 Sun 公司提供的一组技术规格,规划企业用户以 Java2 技术开发、分发、管理多层式应用结构。

微型版(J2ME-Java2 Platform,Micro Edition):适用于消费性电子产品,提供嵌入式系统所使用的 Java 开发工具、执行环境与 API。

2005 年 6 月,JavaOne 大会召开时,Sun 公司公布了 Java SE6。此时,Java 的各种版本已经更名,以取消其中的数字“2”:J2EE 更名为 Java EE,J2SE 更名为 Java SE,J2ME 更名为 Java ME。2009 年 4 月 20 日,Sun 公司被 Oracle 公司(甲骨文公司)收购。本书所有内容基于 Java SE 版本。

## 1.3 Java 开发环境设置

了解 Java 的开发环境是使用 Java 语言编程的第一步,也是学好 Java 语言的基础。目前,Java 开发环境有很多,除 Sun 公司最早提供的免费的 JDK(Java Development Kit,Java

开发工具包)开发环境外,还有常见的 Eclipse、JBuilder、NetBean 等集成开发环境,但都需要提前安装 JDK 工具包。鉴于实际开发中基本都是使用集成开发环境进行开发,故本书仅介绍在 Windows 7 操作系统下的 JDK 与 Eclipse 的安装与使用。

## 1.3.1　下载和安装 JDK

JDK 是整个 Java 的核心,包括 Java 运行环境、Java 工具和 Java 基础类库。掌握 JDK 的安装是学好 Java 的第一步。

1. 下载 JDK

Oracle 公司收购 Sun 公司之后,仍然不断推出新的 JDK 版本,目前最新的版本为 JDK8。读者可以直接访问下面的网址下载 JDK:

http://www.oracle.com/technetwork/java/iavase/downloads/index.html

在下载页面,需要根据自己的操作系统选择合适的 JDK,如图 1-3 所示。

**Java SE Development Kit 8u60**

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

| Product / File Description | File Size | Download |
|---|---|---|
| Linux ARM v6/v7 Hard Float ABI | 77.69 MB | jdk-8u60-linux-arm32-vfp-hflt.tar.gz |
| Linux ARM v8 Hard Float ABI | 74.64 MB | jdk-8u60-linux-arm64-vfp-hflt.tar.gz |
| Linux x86 | 154.66 MB | jdk-8u60-linux-i586.rpm |
| Linux x86 | 174.83 MB | jdk-8u60-linux-i586.tar.gz |
| Linux x64 | 152.67 MB | jdk-8u60-linux-x64.rpm |
| Linux x64 | 172.84 MB | jdk-8u60-linux-x64.tar.gz |
| Mac OS X x64 | 227.07 MB | jdk-8u60-macosx-x64.dmg |
| Solaris SPARC 64-bit (SVR4 package) | 139.67 MB | jdk-8u60-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 99.02 MB | jdk-8u60-solaris-sparcv9.tar.gz |
| Solaris x64 (SVR4 package) | 140.18 MB | jdk-8u60-solaris-x64.tar.Z |
| Solaris x64 | 96.71 MB | jdk-8u60-solaris-x64.tar.gz |
| Windows x86 | 180.82 MB | jdk-8u60-windows-i586.exe |
| Windows x64 | 186.16 MB | jdk-8u60-windows-x64.exe |

图 1-3　JDK 下载页面

2. 安装 JDK

下载完成后就可以安装 JDK 了,安装 JDK 的操作步骤如下:

①双击下载的 JDK 安装文件 jdk-8u31-windows-i586.exe,即可进入 JDK 安装向导界面,如图 1-4 所示。


图 1-4　JDK 安装向导界面

②单击“下一步”按钮进入 JDK 的安装选择界面，如图 1-5 所示。


**图 1-5　JDK 安装选择界面**

③在图 1-5 中，单击“更改”按钮可以选择安装目录，在此保持默认安装目录。在自定义安装程序的功能时，建议选择全部功能。

④单击“下一步”按钮进行安装，在安装过程中，会弹出 JRE(Java 运行环境)的安装提示，如图 1-6 所示，单击“下一步”按钮即可。安装 JRE 过程中会出现如图 1-7 所示的安装界面。


**图 1-6　JRE 安装提示**


**图 1-7　JRE 安装过程**

⑤最后会弹出如图 1-8 所示的界面，单击“完成”按钮，即可完成 JDK 的安装。


**图 1-8　JDK 安装成功**

安装好 JDK 后，JDK 目录下的一些文件和文件夹说明见表 1-1。

**表 1-1　JDK 目录说明**

| 序号 | 文件名称 | 说明 |
|---|---|---|
| 1 | COPYRIGHT | JDK 版本说明文档 |
| 2 | README. html | JDK 的 HTML 说明文档 |
| 3 | README. txt | JDK 基本内容及功能说明文档 |
| 4 | src. zip | JDK 程序源代码压缩文件 |
| 5 | bin 目录 | 包含了常用的 JDK 工具(编译器、解释器等可执行文件) |
| 6 | db 目录 | JDK6-7 附带的一个轻量级的数据库，名字叫作 Derby |
| 7 | include 目录 | 包含了一些与 C 程序连接时所需的文件 |
| 8 | jre 目录 | 存放 Java 运行环境文件 |
| 9 | lib 目录 | 存放 Java 的类库文件 |

3. 设置环境变量

JDK 安装成功后，还需要对操作系统的环境变量进行设置。

①在 Windows 操作系统桌面上右击“计算机”图标，在弹出的快捷菜单中选择“属性”命令，弹出“系统属性”对话框，切换到“高级”选项卡，如图 1-9 所示。

②单击“环境变量”按钮，打开“环境变量”对话框，如图 1-10 所示。


**图 1-9　“系统属性”对话框**


**图 1-10 “环境变量”对话框**

③在“环境变量”对话框中单击“系统变量”选项组下方的“新建”按钮，在弹出的“新建系统变量”对话框中输入变量名“JAVA_HOME”，用于指定 JDK 的位置，其变量值为“C.\Program Files\Java\jdkl.8.0_31”，即 JDK 的安装目录，如图 1-11 所示，然后单击“确定”按钮即可。

④按照同样的方法，新建系统环境变量“CLASSPATH”，用于 Java 加载类(Class 或 lib)的路径，其变量值为“.;%JAVA_HOME%\lib”，其中“.”不能少，它表示当前目录。单击“确定”按钮完成设置，如图 1-12 所示。


**图 1-11 “新建系统变量”对话框**


**图 1-12 新建 CLASSPATH 系统变量**

⑤在“系统变量”选项组中选择 Path 选项，用于安装路径下识别 Java 命令。单击其下方的“编辑”按钮，弹出“编辑系统变量”对话框，在当前变量值的基础上，增加“;%JAVA_HOME%\bin”，如图 1-13 所示。


**图 1-13 编辑 Path 系统变量**

⑥安装并配置好 JDK 之后，选择“开始”→“cmd”命令，打开 DOS 窗口。在 DOS 窗口分别输入 Javac 和 Java 命令，如果能看到如图 1-14 和图 1-15 所示的提示信息，则说明安装正确，否则需要重新设置环境变量。

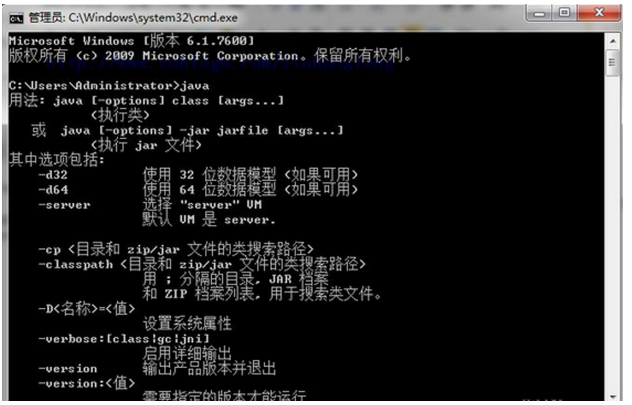



图 1-14　Java 命令提示信息(1)

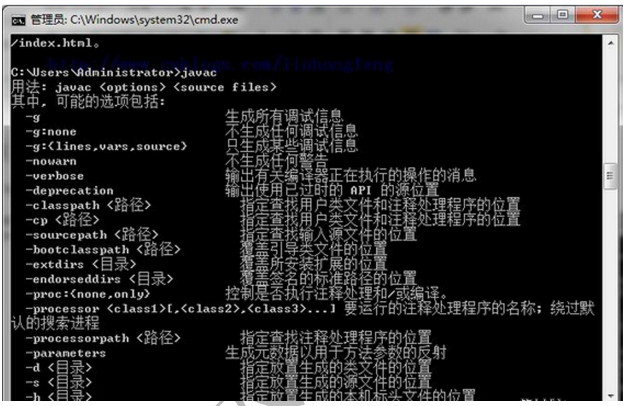



图 1-15　Java 命令提示信息(2)

## 1.3.2　下载和安装 Eclipse

Eclipse 是一个开放的可扩展的集成开发环境，不仅可用于 Java 桌面程序的开发，而且可以通过安装开发插件构建 Web 项目等的开发环境。Eclipse 是开放源代码的项目，可以免费下载。

1. 下载安装 Eclipse

①在 Eclipse 的官方网址 http://www.eclipse.org 首页上找到下载栏目，下载最新版本的 eclipse-java-luna-SRla-win32.zip。

②解压 eclipse-java-luna-SRla-win32.zip 到一个目录，例如解压到 D:\下面，则会生成一个 D:eclipse 文件，这个是 eclipse 的文件夹。

2. Eclipse 界面说明

单击 eclipse.exe，运行 Eclipse 集成开发环境。在第一次运行时，Eclipse 会要求选择工作空间(workspace)，用于存储工作内容(本书选择 D:\workspace 作为工作空间)，如图 1-16 所示。

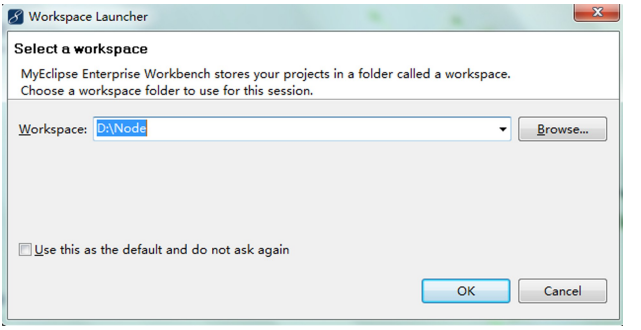



**图 1-16　Eclipse 选择工作空间**

选择工作空间后，Eclipse 打开工作空间，如图 1-17 所示。转至工作台窗口后，Eclipse 界面提供了一个或多个透视图，透视图包含编辑器和视图（如导航器）。用户可同时打开多个工作台窗口。

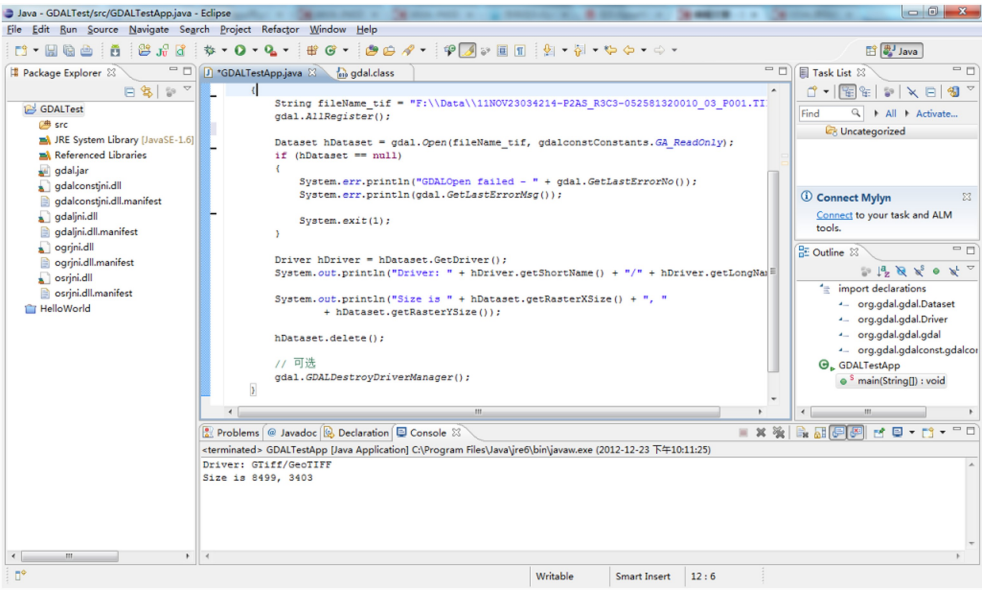



**图 1-17　Eclipse 工作台**

Eclipse 工作台由几个被称为视图（view）的窗格构成，窗格的集合称为透视图（perspective）。Java 透视图包含一组更适合于 Java 开发的视图。

该工作台有一个便利的特性，即可自定义工作台。使用“Window”菜单下的“Reset Perspective”（复位透视图），可将布置还原成程序初始状态；也可以从“Window”菜单下的“Show View”（显示视图）中选取一个视图来显示。

## 同步思考

1. 为什么安装了 JDK 却不能使用 javac 和 java 命令来运行程序？
2. 怎样才能学好 Java 语言？
3. 编写一个 Java Application 程序来输出自己的学号与名字。
4. 编写 Java Application 程序，输出一个由 6 行“*”组成的直角三角形。

# 第 2 章　数据类型和运算符

**本章导读**

程序是由文档、数据和处理这些数据的算法组成的。数据及其运算是任何一门语言所必需的，Java 语言也不例外。在中小学数学中有整数、小数等概念的存在，那么 Java 作为一门计算机语言，在计算机中如何进行数据的表达和识别呢？为了解决这个问题，Java 语言和别的高级语言一样，引用了数据类型，利用数据类型声明来完成。

**知识要点**

- Java 的标识符、关键字和数据类型的使用
- 通过命令进行数据输入输出的方法
- 运算符、表达式和各种语句的使用



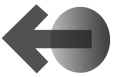
## 2.1　数 据 类 型

### 2.1.1　关键字和标识符

1. 标识符

标识符是指可被用来为类、变量或方法等命名的字符序列。Java 语言规定标识符由字母、数字、下划线和美元符号（$）组成，并且第一个字符不能是数字。Java 是大小写敏感的语言，即标识符中的字符是区分大小写的，如 age 和 Age 代表着不同的标识符。

Java 中所谓的字母并不只包含英文字母、数字及一些常用符号。需要特别注意的是，Java 采用的为 Unicode 字符集。Unicode 字符集是一个由名为 Unicode 学术学会（Unicode Consortium）的机构制订的字符编码系统，它为世界上 650 多种语言中的每个字符设定了统一并且唯一的编码，以满足跨语言、跨平台进行文本转换和处理的要求。Unicode 标准始终使用十六进制数字编码，其中前 128 个字符与 ASCII 码表中的字符对应。到 Unicode 4.0



规范为止，有 96 382 个码位被分配了字符，而汉字的编码在 Unicode 1.0 版本中已有 20 902 个，到了 2012 年的 6.1 版本中扩展到 74 617 个汉字，如汉字中的“礼”字就是 Unicode 字符集中的第 31 036 个字符。因此，Java 可使用的字符不仅可以是英文字母等，也可以是汉字、朝鲜文、俄文、希腊字母以及其他许多语言中的文字。例如以下都是合法的标识符：

year、numl、$cost、_page、LEAP_NUM、按钮 1

2. 关键字

关键字(keyword)是由系统定义的一些字符串，代表某些特定的含义，Java 中定义了 50 个关键字，详见表 2-1。

**表 2-1　Java 关键字**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| abstract | assert | boolean | break | byte | case | catch | char |
| class | const | contmue | default | do | double | else | enum |
| extends | final | finally | float | for | goto | if | implements |
| maport | instanceof | int | interface | long | native | new | package |
| private | protected | public | return | short | static | strictfp | super |
| switch | synchronized | this | throw | throws | transient | try | void |
| volatile | while | | | | | | |

在 Java 中还有保留字(reserred word)，是为后续版本预留的，虽然现在未被使用，但在以后的版本中有可能作为关键字。其中，const 和 goto 虽为关键字，但未被使用，它们被用作保留字，以备扩充。strictfp 是从 1.2 版本起开始使用，assert 和 enum 则分别是在 1.4 和 5.0 版本中引入的。除此之外，还有 3 个用于表示特殊值的保留字：true、false 和 null。现在 Java 中的保留字包括：byValue、cast、false、future、generic、inner、operator、outer、rest、true、var、goto、const、null。

3. 注释

为程序添加注释可以解释某些语句的作用和功能，提高程序的可读性。同时在注释中可以添加编程人员的个人信息。此外，注释可以屏蔽一些不要执行的语句，在编译的时候不会执行注释，在需要执行的时候可以去掉注释。注释功能可以分为以下 3 类。

(1)单行注释。

单行注释只需要在注释内容的前面添加“//”就可以了。如下面的例子。

```
int a=13;//定义变量 a,同时赋值为 13
```

(2)多行注释。

多行注释就是在注释内容的前面添加一个“/*”，并且在注释的末尾添加一个“*/”。例子如下。

```
/*
  int a=14;
  int b=15;
*/
```

(3)文档注释。

文档注释是以单斜线外加两个星行标记开头(/**),以星标记加单斜线(*/)结束。文档注释的内容会被编译成程序的正式文档,并能包含在 javadoc 类的工具生成文档里,从而进行说明文档的层次和结构。

## 2.1.2 数据类型

Java 中的数据类型分为两种:基本类型和引用数据类型。基本类型又称为原始数据类型、简单类型,是不能简化的、内置的数据类型,由编程语言本身定义。引用数据类型也称为复合类型、扩展类型。Java 语言本身不支持 C++中的结构(struct)或联合(union)数据类型,它的复合数据类型一般都是通过类或接口进行构造,类提供了捆绑数据和方法的方式,同时可以针对程序外部进行信息隐藏。

Java 基本类型共有 8 种,可以分为两类,即布尔类型 boolean 以及数值类型 byte、short、int、long、float、double、char(char 本质上是一种特殊的 int)。Java 中的数值类型不存在无符号的,它们的取值范围是固定的,不会随着机器硬件环境或者操作系统的改变而改变。

Java 有一种表示逻辑值的简单类型,称为布尔型。布类型代表逻辑中的成立和不成立。Java 语言中使用保留字 true 代表成立,false 代表不成立。布尔型数据只有这两个值,且它不对应于任何整数值,在流控制中常用到它。

2. 整数型

整数型是一类代表整数值的类型。Java 中有字节型、整型、短整型和长整型 4 种。当需要代表一个整数的值时,可以根据需要从 4 种类型中挑选合适的,如果没有特殊要求的话,一般选择整型。4 种整数型的区别主要在每个数据在内存中占用的空间大小和代表的数值的范围。具体说明参见表 2-2。另外,不像 C/C++,Java 不支持无符号类型(unsigned)。

3. 浮点型

浮点型是一类代表小数值的类型。由于小数的存储方式和整数不同,所以小数都有一定的精度,所以在计算机中运算时不够精确。根据精度和存储区间的不同,设计了两种小数类型,可以根据需要从中挑选合适的。如果没有特殊要求,一般选择 double 类型。具体见表 2-2。

4. 字符型

字符型代表特定的某个字符,计算机中都是以字符集的形式来保存字符的,所以字符型的值实际只是字符集中的编号,而不是实际代表的字符,由计算机完成从编号转换成对应字符的工作。但是字符型的编号中不包含负数,且由于字符型数据存储的是编号的数值,所以可以参与数学运算。字符型可以作为 Java 语言中的无符号整数使用。最后需要强调的是字符型的默认值是编号为 0 的字符,而不是字符“0”。

表 2-2　基本数据类型

| 类型 | 关键字 | 大小(bit) | 取值范围 | 默认值 |
|---|---|---|---|---|
| 布尔型 | boolean | 1 | false 或 true | false |
| 字节型 | byte | 8 | −128～127 | 0 |
| 短整型 | short | 16 | −32768～32767 | 0 |
| 整型 | int | 32 | $-2^{31}$～$2^{31}-1$ | 0 |
| 长整型 | long | 64 | $-2^{63}$～$2^{63}-1$ | 0L |
| 单精度浮点型 | float | 32 | 大约$\pm 3.4\times10^{38}$ | 0.0f |
| 双精度浮点型 | double | 64 | 大约$\pm 1.7\times10^{308}$ | 0.0d |
| 字符型 | char | 16 | 16 位 Unicode | '10' |

## 2.2　常量与变量



### 2.2.1　常量

在程序运行中其值不能被改变的一类数据称为常量。常量主要有两个作用:代表常数,便于程序的修改;增强程序的可读性。

1. 整型常量

整型常量有 3 种表示方式:十进制、八进制、十六进制。

十进制整型常量:如 418、−316、0。

八进制整型常量:以 0 开头的整数。如 0101 表示十进制数 65、−021 表示十进制数 −17。

十六进制整型常量:以 0x 或 0X 开头,如 0x101 表示十进制数 257、−0X21 表示十进制数 −33。

一个整型常量被当作是 int 类型,不管实际数值有多大,并且在一个整型常数后面加上字母“L”(大小写均可,但一般为了避免小写字母 l 与数字字符 1 混淆,建议使用大写字母 L),如 12345L,则代表长整数。

从 Java 7 开始,可以使用前缀 0b 来表示二进制数据,例如 0b1001 对应十进制中的 9。同样从 Java 7 开始,可以使用下划线来分隔数字,类似英文数字写法,例如 1 000 000 表示 1000000,也就是一百万。下划线只是为了让代码更加易读,编译时编译器会删除这些下划线。

2. 浮点型常量

浮点型常量有小数计数法和科学计数法两种形式。

小数计数法:由数字和小数点组成,如 0.456、.456、456.、456.0。

科学计数法：由一般实数和 e±n(E±n)组成，如 45.6e3、2E-8，它们分别表示 45.6 乘以 10 的 3 次方，2 乘以 10 的－8 次方。特别的，e 或 E 之前必须有数字，之后必须为整数。

**注意**：浮点型常量的默认类型是 double 型，所以 float 类型的后面一定要加 f(F)。同样带小数的变量默认为 double 类型。float 类型有效数字最长为 7 位，有效数字长度包括了整数部分和小数部分。double 类型有效数字最长为 15 位。

3. 字符常量

字符常量指用单引号括起来的单个字符，如'a'、'A'、'单'、'双'。注意，使用的是单引号，而非双引号。除了以上所述形式的字符常量之外，Java 还允许使用一种特殊形式的字符常量值，这通常用于表示难以用一般字符来表示的字符，这种特殊形式的字符是以一个"\"开头的字符序列，称为转义字符。与 C、C＋＋不同，Java 中的字符型数据是 16 位无符号型数据，使用的为 Unicode 字符集，而不仅仅是 ASCII 集。Java 中的常用转义字符如表 2-3 所示。

**表 2-3　Java 中的常用转义字符**

| 转义字附 | 描述 |
|---|---|
| \ddd | 1 到 3 位八进制数据所表示的字符(ddd) |
| \uxxxx | 1 到 4 位十六进制数所表示的字符(xxxx) |
| \' | 单引号字符(\u0027) |
| \" | 双引号字符(\u0022) |
| \\ | 反斜杠字符(\u005C) |
| \r | 回车(\u000D) |
| \n | 换行(\u000A) |
| \f | 走纸换页(\u000C) |
| \t | 水平制表符(\u0009) |
| \b | 退格(\u0008) |

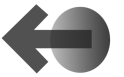


## 2.2.2　变量

在程序运行过程中，有些数据的值会发生改变，这类数据被称为变量。计算机界一代宗师 E. W. Dijkstra 说过一句话："如果理解了编程时变量的用法，也就理解了编程的精粹。"Java 的变量是作为存储单元来实现的，对每个变量，编译器都会分配一部分存储单元用来存储变量的值。

声明变量时，首先要让计算机知道变量的名字。还需要使计算机知道需要为这个变量预留多大的内存空间，因为不同类型的变量需要的内存空间是不同的。例如，在求圆面积的程序中，需要表示半径 r 和面积 area 这两个变量，因为对于不同的圆，这两个变量的值会发生改变。

(1)由于 Java 语言是一种强类型的语言，所以变量在使用以前必须首先声明，在程序中



声明变量的语法格式如下。

```
数据类型　变量名称;
```

例如:int a;

在该语法格式中,数据类型可以是 Java 语言中任意的类型,包括前面介绍到的基本数据类型以及后续将要介绍的复合数据类型。变量名称是该变量的标识符,需要符合标识符的命名规则。数据类型和变量名称之间使用空格进行间隔,空格的个数不限,但是至少需要 1 个。语句使用“;”作为结束。

(2)也可以在声明变量的同时,对变量进行初始化,语法格式如下。

```
数据类型　变量名称=值;
```

例如:int num=5;

在该语法格式中,前面的语法和上面介绍的内容一致,后续的“=”代表赋值,其中的“值”代表具体的数据。在该语法格式中,要求值的类型和声明变量的数据类型一致。

(3)也可以一次声明多个相同类型的变量,语法格式如下。

```
数据类型　变量名称 1,变量名称 2,…,变量名称 n;
```

例如:int num,score,age;

在该语法格式中,变量名之间使用“,”分隔,这里的变量名称可以有任意多个。

(4)也可以在声明多个变量时对变量进行赋值,语法格式如下。

```
数据类型　变量名称 1=值 1,变量名称 2=值 2,…,变量名称 n=值 n;
```

例如:int num=5,score=80,age=19;

(5)也可以在声明变量时,进行部分变量的赋值。

例如:int num,score=80,age;

以上语法格式中,如果同时声明多个变量,则要求这些变量的类型必须相同。如果声明的变量类型不同,则只需要分开声明即可,例如:

```
int num=3;
boolean flag=false;
char ch;
```

在程序中,变量的值代表程序的状态,在程序中可以通过变量名称来引用变量中存储的值,也可以为变量重新赋值。例如:

```
int num=5;
num=10;
```

**【例 2-1】** 不同数据类型变量操作。

```
public class Exam2_1{
    public static void main(String[]args){
     //字符型
     char Name1='张';
     char Name2='三';
```

```
        System.out.println("学生的姓名为:"+Name1+Name2);
        //整数型
        short x=30;          //十进制
        int y=030;           //八进制
        long z=0x30L;        //十六进制
        System.out.println("转化成十进制:x="+x+",y="+y+",z="+z);
        //浮点型
        float m=12.345f;
        double n=10;
        System.out.println("计算乘积:"+m+"*"+n+"="+m*n);
        //布尔型
        boolean flagI=false,flag2=true;
        System.out.println("逻辑值 flag1:"+flag1+",逻辑值 flag2:"+flag2);
    }
}
```

运行结果如下：

学生的姓名为:张三

转化成十进制数:x=30,y=24,z=48

计算乘积:12.345 * 10.0=123.45000267028809

逻辑值 flag1:false,逻辑值 flag2:true

从运行结果可以看出，即使浮点型数据只有整数没有小数，在控制台上输出时系统也会自动加上小数点，并且小数位全部置为 0。

**注意**:(1)boolean 型变量未初始化时，默认值为 false。除此以外，boolean 型变量不同于其他的基本数据类型，它不能被转换成任何其他的基本类型，其他的基本类型也不能被转换成 boolean 类型。

(2)在变量声明的时候，在类型的前边使用 final 修饰，表示声明的是一个常量，例如：

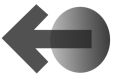
```
final int I=10;
final float PI=3.1415926;
```

由 final 修饰的常量在声明时必须初始化。

除基本数据类型外，final 可以修饰任何数据类型的量，使其成为常量。

## 2.3 运 算 符

程序最基本的功能就是计算。但不同于普通数学公式中的运算符号，在 Java 中的运算符，基本上可分为算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符等。运算符



与操作数组合成表达式。

## 2.3.1　赋值运算符

赋值使用运算符“=”。它的意思是“取右边的值(即右值),把它赋值给左边(即左值)”。右值可以是任何常数、变量或者表达式(只要它能生成一个值就行),但左值必须是一个明确的、已命名的变量。也就是说,必须有一个物理空间可以存储等号右边的值。赋值运算符是程序中最常用的运算符了,只要有变量的声明,就要有赋值运算。

对基本数据类型的赋值是很简单的。基本数据存储了实际的数值,而并非指向一个对象的引用,所以在为其赋值的时候,是直接将一个地方的内容复制到了另一个地方。

但是在为对象“赋值”的时候,情况却有了变化。对一个对象进行操作,真正操作的是对象的引用。这种特殊的现象通常称作“别名现象”,是 Java 操作对象的一种基本方式。详细内容会在后续章节提到。

## 2.3.2　算术运算符

算数运算符号中,加、减、乘的结果都是非常容易理解的,重点讲一下除(/)的运算和取余(%)运算。Java 程序中,两个整数相除的结果是整数,这与普通数学运算是不同的,求出的结果会直接截取小数部分(不会四舍五入)。而在 Java 中,两个整数求余的结果类似于数学中的求模运算。

**【例 2-2】**　两个整数相除及求余数。

```
public class Exam2_2 {
    public static void main(String[] args) {
        int a=-15,x=15;
        int b=2,y=-2;
        double c=2;
        System.out.println(a+"/"+b+"="+(a/b));
        System.out.println(a+"%"+b+"="+ (a%b));
        System.out.println(x+"/"+y+"="+ (x/y));
        System.out.println(x+"%"+y+"="+(x%y));
        System.out.println(a+"/"+y+";"+ (a/y));
        System.out.println(a+"%"+y+"="+(a%y));
        System.out.println(a+"/"+C+"="+ (a/c));
        System.out.println(a+"%"+c+"="+ (a%c));
    }
}
```

输出结果如下:

```
-15/2=-7
```

－15％2＝－1

15/－2＝－7

15％－2＝1

－15/－2＝7

－15％－2＝－1

－15/2.0＝－7.5

－15％2.0＝－1.0

通过以上的例子可以看出，求余运算结果的正负号与被除数的正负号一致。

在循环与控制中，经常会用到类似于计数器的运算，它们的特征是每次的操作都是加 1 或减 1。在 Java 中提供了自增、自减运算符，x＋＋、＋＋x 均可使变量 x 的当前值每次增加 1，区别在于后缀式(x＋＋)的自增运算在作为表达式的一部分时，会先用 x 的值参与运算，而后再对 x 变量进行自加；而前缀式则是先对 x 变量进行自加，x 加 1 后的结果再作为表达式的一部分参与运算。自减运算符和自增运算符的操作方法类似，只不过做的是减法而已。

**【例 2-3】** 自增、自减运算符的使用。

```
public class Exam2_3{
    public static void main(String[] args) {
      int x=10,y=2;
      System.out.println("x="+(x++));
      System.out.println("x="+(++x));
      System.out.println("y="+(y--));
      System.out.println("y="+(--y));
    }
}
```

输出结果如下：

x＝10

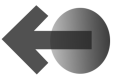
x＝12

y＝2

y＝0

## 2.3.3 关系运算符

数学上有比较的运算，如大于、等于、小于等运算。Java 中也提供了这些运算符，它们被称为“关系运算符”(Comparison Operator)，其中有大于(＞)、大于等于(＞＝)、小于(＜)、小于等于(＜＝)、等于(＝＝)和不等于(！＝)。在 Java 中，关系条件成立时以 boolean 类型的值 true 表示，不成立时以值 false 表示，而不是 C 或 C＋＋中的 1 或 0。Java 中，任何数据类型的数据(包括基本类型和组合类型)都可以通过＝＝或！＝来比较是否相等(这与 C、C＋＋不同)。



【例 2-4】　关系运算符的使用。

```
public class Exam2_4 {
    public static void main(String[] args) {
        int a=15;
        int b=31;
        int c=15;
        System.out.println("a>b 的结果为"+(a>b));
        System.out.println("a<b 的结果为"+(a<b));
        System.out.println("a==c 的结果为"+(a==c));
    }
}
```

输出结果如下：

a>b 的结果为 false

a<b 的结果为 true

a==c 的结果为 true

## 2.3.4　位运算符

所有的数据、信息在计算机中都是以二进制形式存在的。可以对整数的二进制位进行相关的操作，这就是位运算符的作用。它主要包括：位的“与”、位的“或”、位的“非”和位的“异或”。

1. 位的“与”

位的“与”用符号“&”表示，它属于二元运算符。“与运算”规则如表 2-4 所示。

“与运算”的特殊用途如下。

(1)清零。如果想将一个数据清零，即使其全部二进制位为 0，只要与一个各位都为零的数值相与，结果即为零。

(2)取一个数中指定的位。

方法：找一个数，对应 X 要取的位，该数的对应位为 1，其余位为零，此数与 X 进行“与运算”可以得到 X 中的指定位。

例：设 X=10101110，取 X 的低 4 位，用 X&0000 1111=0000 1110 即可得到；还可用来取 X 的 2、4、6 位。

2. 位的“或”

位的“或”用符号“|”表示，它属于二元运算符。“或运算”规则如表 2-4 所示。

“或运算”常用来对一个数据的某些位置 1。

方法：找到一个数，对应 X 要置为 1 的位，该数的对应位为 1，其余位为零。此数与 X 相或可使 X 中的某些位置 1。

例：将 X=10100000 的低 4 位置 1，用 X|0000 1111=1010 1111 即可得到。

3. 位的“非”

位的“非”用符号“～”表示，它是一元运算符，只对单个自变量起作用。它的作用是使二进制按位“取反”。“非运算”规则如表 2-4 所示。

4. 位的“异或”

位的“异或”用符号“^”表示，它属于二元运算符。“异或运算”规则如表 2-4 所示。

“异或运算”的特殊作用如下。

(1)使特定位翻转。找一个数，对应 x 要翻转的各位，该数的对应位为 1，其余位为零，此数与 x 对应位异或即可。

例：X＝10101110，使 X 低 4 位翻转，用 X^0000 1111＝1010 0001 即可得到。

(2)与 0 相异或，保留原值，X^0000 0000＝1010 1110。

**表 2-4　位运算规则表**

| a | b | a&b | a\|b | ～a | a^b |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |



**【例 2-5】** 位运算操作运算实例。

```
public class Exam2_5 {
    public static void main(String[] args){
        int a=129;
        int b=128;
        int x=a&b;
        int y=a|b;
        int z=a^b;
        int t=～a;
        System.out.println(a+"&"+b+"="+x);
        System.out.println(a+"|"+b+"="+y);
        System.out.println(a+"^"+b+"="+z);
        System.out.println("～"+a+"="+t);
    }
}
```

输出结果如下：

129&128＝128129|128＝129

129^128＝1

～129＝126

a 的值是 129，转换成二进制数就是 10000001，而 b 的值是 128，转换成二进制数就是

10000000。根据与运算符的运算规律，只有两个位都是 1，结果才是 1，可以知道结果就是 10000000，即 128；根据或运算符的运算规律，只有两个位有一个是 1，结果才是 1，可以知道结果就是 10000001，即 129；根据异或运算符的运算规律，两位相异结果才是 1，可以知道结果就是 00000001，即 1；而对 129 的二进制按位取反得到的二进制数为 01111110，即 126。

在使用位运算符时还需要注意以下问题。

(1)负数按补码形式参加按位与运算。

(2)若进行位逻辑运算的两个操作数的数据长度不相同，则返回值应该是数据长度较长的数据类型。

(3)按位异或可以实现不使用临时变量完成两个值的交换，即 a=a^b;b=a^b;a=a^b。

除了以上 4 种位运算符外，Java 中还有一类移位运算符，其操作对象也是二进制的“位”。可以单独用移位运算符来处理 int 型数据。它主要包括：左移位运算符(<<)、“有符号”右移位运算符(>>)和“无符号”右移运算符(>>>)。

5. 左移位运算符

左移位运算符，用符号“<<”表示。它是将运算符左边的对象向左移动运算符右边指定的位数(在低位补 0)。

6. “有符号”右移运算符

“有符号”右移运算符，用符号“>>”表示。它是将运算符左边的运算对象向右移动运算符右侧指定的位数。它使用了“符号扩展”机制，也就是说，如果值为正，在高位补 0；若为负，则在高位补 1。

7. “无符号”右移运算符

“无符号”右移运算符，用符号“>>>”表示。它同“有符号”右移运算符的移动规则是一样的，唯一的区别就是：“无符号”右移运算符采用了“零扩展”，也就是说，无论值为正还是为负，都在高位补 0。

**【例 2-6】** 移位运算符操作。

```
public class Exam2_6 {
    public static void main(String[] args) {
        int a=31;
        int b=3;
        int x=a<<b;
        int y=a>>b;
        int z=a>>>b;
        System.out.println(a+"<<"+b+"="+x);
        System.out.println(a+">>"+b+"="+y);
        System.out.println(a+">>>"+b+"="+z);
    }
}
```

输出结果如下：

31<<2=248

31>>2=3

31>>>2=3

## 2.3.5 逻辑运算符

在 Java 语言中有 3 种逻辑运算符，它们是 NOT(非，以符号“!”表示)、AND(与，以符号“&&”表示)、OR(或，以符号“||”表示)。具体规则如表 2-5 所示。

**表 2-5 逻辑运算规则表**

| A | B | ! A | A&&B | A||B |
|---|---|---|---|---|
| true | true | false | true | true |
| false | true | true | false | true |
| true | false | false | false | true |
| false | false | true | false | false |

**【例 2-7】** 逻辑运算符操作。

```
public class Exam2_7 {
    public static void main(String[] args) {
        boolean x,y,z,a,b;
        a='c'>'C';
        b='r'! ='r';
        x=! a;
        y=a&&b;
        z=a||b;
        System.out.println("x="+x);
        System.out.println("y="+y);
        System.out.println("z="+z);
    }
}
```

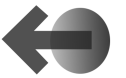
输出结果如下。

x=false

y=false

z=true

“&&”运算符检查第一个表达式是否返回 false，如果是 false 则结果必为 false，不再检查其他内容。“||”运算符检查第一个表达式是否返回 true，如果是 true 则结果必为 true，不再检查其他内容。这就像电路中的短路一样，因此“&&”和“||”又被称为短路运算符。



## 2.3.6　其他运算符

1. 复合赋值运算符

在赋值运算符当中，还有一类复合赋值运算符。它们实际上是一种缩写形式，使得对变量的改变更为简洁。

例如：total＝total＋3；简化写成 total＋＝3；

那么这两种写法有区别吗？答案是有的，例如对于 total＝total＋3，表达式 total 被计算了两次，对于复合运算符 total＋＝3，表达式 total 仅计算了一次。一般来说，这种区别对于程序的运行没有多大影响，但是当表达式作为函数的返回值时，函数就被调用了两次，而且如果使用普通的赋值运算符，也会加大程序的开销，使效率降低。复合赋值运算如表 2-6 所示。

**表 2-6　复合赋值运算一览表**

| 运算符 | 一般表示法 | Java 语言表示法 |
|---|---|---|
| ＋＝ | a＝a＋b | a＋＝b |
| －＝ | a＝a－b | a－＝b |
| ＊＝ | a＝a＊b | a＊＝b |
| /＝ | a＝a/b | a/＝b |
| %＝ | a＝a%b | a%＝b |
| ＞＞＝ | a＝a＞＞b | a＞＞＝b |
| ＞＞＞＝ | a＝a＞＞＞b | a＞＞＞＝b |



2. 条件运算符

条件运算符比较罕见，因为它有 3 个运算对象，一般形式为：表达式 1？表达式 2：表达式 3。

以下是关于条件运算符的几点说明。

(1)通常情况下，表达式 1 是关系表达式或逻辑表达式，用于描述条件表达式中的条件，表达式 2 和表达式 3 可以是常量、变量或表达式。

(2)条件表达式的执行顺序为：先求解表达式 1，若值为 true，表示条件为真，则求解表达式 2，此时表达式 2 的值就作为整个条件表达式的值；若表达式 1 的值为 false，表示条件为假，则求解表达式 3，表达式 3 的值就是整个条件表达式的值。例如：

```
(a>=0)? a:-a
```

执行结果是 a 的绝对值。

(3)条件表达式的优先级别仅高于赋值运算符，而低于前面遇到过的所有运算符。

(4)条件表达式允许嵌套，即允许条件表达式中的表达式 2 和表达式 3 又是一个条件表达式。例如：

x>0? 1:x<0? -1:0 等价于 x>0? 1:(x<0? -1:0)

(5)表达式 1,表达式 2,表达式 3 的类型可以不同。此时条件表达式的值的类型为它们中精度较高的类型。

3. 逗号运算符

在 Java 中,多个表达式可以用逗号分开,其中用逗号分开的表达式的值分别结算,但整个表达式的值是最后一个表达式的值。在 Java 中,逗号运算符的唯一使用场所就是在 for 循环语句中。

4. 字符串运算符

“+”号这个运算符,在 Java 中有一项特殊的用法,它不仅起到连接不同的字符串的作用,还有一种隐式的转型功能。

## 2.3.7 优先级和结合性

最后总结一下运算符的优先级以及结合性,如表 2-7 所示。

**表 2-7 运算符优先级和结合性**

| 运算符 | 优先级 | 结合性 |
|---|---|---|
| 0[].  | 1(最高) | 从左到右 |
| ! +(正)-(负)~++-- | 2 | 从右向左 |
| */% | 3 | 从左向右 |
| +(加)-(减) | 4 | 从左向右 |
| <<>>>>> | 5 | 从左向右 |
| <<=>>=installceof | 6 | 从左向右 |
| ==!= | 7 | 从左向右 |
| &(按位与) | 8 | 从左向右 |
| ^ | 9 | 从左向右 |
| \| | 10 | 从左向右 |
| && | 11 | 从左向右 |
| \|\| | 12 | 从左向右 |
| ?: | 13 | 从右向左 |
| =+=-=*=/=%=&=\|=^=~=<<= >>=>>>= | 14 | 从右向左 |

其实在实际的开发中,不需要去记忆运算符的优先级别,也不要刻意地使用运算符的优先级别,对于不清楚优先级的地方使用小括号进行替代。

# 2.4　Java 数据类型的转换

Java 语言中的数据类型转换有以下两种。

自动类型转换：编译器自动完成类型转换，不需要在程序中编写代码。

强制类型转换：强制编译器进行类型转换，必须在程序中编写代码。

由于基本数据类型中 boolean 类型不是数字型，所以基本数据类型的转换是除了 boolean 类型以外的其他 7 种类型之间的转换。

## 2.4.1　自动类型转换

自动类型转换，也称隐式类型转换，是指不需要书写代码，由系统自动完成的类型转换。由于实际开发中这样的类型转换很多，所以 Java 语言在设计时，没有为该操作设计语法，而是由 JVM 自动完成。从存储范围小的类型到存储范围大的类型。即：

byte→short(char)→int→long→float→double

例如：

```
byte b=50;
char c='a';
short s=1024;
int i=50000;
float f=5.67f;
double d=0.1234;
double result=(f*b)+(i/c)-(d*s);
```

第 1 个表达式 f * b 中，b 被提升为 float 类型，该子表达式的结果也提升为 float 类型。

第 2 个表达式 i/c 中，变量 c 被提升为 int 类型，该子表达式的结果提升为 int 类型。

第 3 个表达式 d * s 中，变量 s 被提升为 double 类型，该子表达式的结果提升为 double 型。

最后，这 3 个子表达式的结果类型分别是 float、int 和 double，相减后该表达式的最终的结果就是 double 类型。

**注意**：byte、short、char 之间不会互相转换，并且三者在计算时首先转换为 int 类型。

## 2.4.2　强制类型转换

强制类型转换，也称显式类型转换，是指必须书写代码才能完成的类型转换。这种类型转换很可能存在精度的损失。强制类型转换的格式如下：

(强制转换类型)变量名称

**【例 2-8】** 强制类型转换实例。

```
public class Exam2_8 {
    public static void main(String[] args) {
        int x;
        double y;
        X=(int)12.5+(int)67.7;//强制转型可能引起精度丢失
        y=(double)x;
        System.out.println("x="+x);
        System.out.println("y="+y);
    }
}
```

输出结果如下：

x=79

y=79.0

对 byte、short、char 三种类型而言，它们是平级的，不能相互自动转换，可使用强制类型转换。例如：

```
byte b1=(byte)short1;char c1=(char)short1;
```

## 同步思考

1. 设 x=2，则表达式(x++)/3 的值是多少？
2. 若 x=5，y=10，则 x<y 和 x>=y 的逻辑值分别是什么？
3. 定义类的关键字是什么？定义接口的关键字是什么？